New Sheet

Operation codes

E.MUL.ADD. 8	Ensemble multiply signed bytes add doublets
E.MUL.ADD.16	Ensemble multiply signed doublets add quadlets
E.MUL.ADD.32	Ensemble multiply signed quadlets add octlets
E.MUL.ADD.64	Ensemble multiply signed octlets add hexlet
E.MUL.ADD.C. 8	Ensemble multiply complex bytes add doublets
E.MUL.ADD.C.16	Ensemble multiply complex doublets add quadlets
E.MUL.ADD.C.32	Ensemble multiply complex quadlets add octlets
E.MUL.ADD.M. 8	Ensemble multiply mixed-signed bytes add doublets
E.MUL.ADD.M.16	Ensemble multiply mixed-signed doublets add quadlets
E.MUL.ADD.M.32	Ensemble multiply mixed-signed quadlets add octlets
E.MUL.ADD.M.64	Ensemble multiply mixed-signed octlets add hexlet
E.MUL.ADD.U. 8	Ensemble multiply unsigned bytes add doublets
E.MUL.ADD.U.16	Ensemble multiply unsigned doublets add quadlets
E.MUL.ADD.U.32	Ensemble multiply unsigned quadlets add octlets
E.MUL.ADD.U.64	Ensemble multiply unsigned octlets add hexlet
E.MUL.SUB. 8	Ensemble multiply signed bytes subtract doublets
E.MUL.SUB.16	Ensemble multiply signed doublets subtract quadlets
E.MUL.SUB.32	Ensemble multiply signed quadlets subtract octlets
E.MUL.SUB.64	Ensemble multiply signed octlets subtract hexlet
E.MUL.SUB.C. 8	Ensemble multiply complex bytes subtract doublets
E.MUL.SUB.C.16	Ensemble multiply complex doublets subtract quadlets
E.MUL.SUB.C.32	Ensemble multiply complex quadlets subtract octlets
E.MUL.SUB.M. 8	Ensemble multiply mixed-signed bytes subtract doublets
E.MUL.SUB.M.16	Ensemble multiply mixed-signed doublets subtract quadlets
E.MUL.SUB.M.32	Ensemble multiply mixed-signed quadlets subtract octlets
E.MUL.SUB.M.64	Ensemble multiply mixed-signed octlets subtract hexlet
E.MUL.SUB.U. 8	Ensemble multiply unsigned bytes subtract doublets
E.MUL.SUB.U.16	Ensemble multiply unsigned doublets subtract quadlets
E.MUL.SUB.U.32	Ensemble multiply unsigned quadlets subtract octlets
E.MUL.SUB.U.64	Ensemble multiply unsigned octlets subtract hexlet

Selection

class	op	type	prec
multiply	E.MUL.ADD	NONE M U	8 16 32 64
complex multiply	E.MUL.SUB	C	8 16 32

FIG. 54A

New Sheet

Format

E.op.size rd@rc,rb

rd=gopsize(rd,rc,rb)

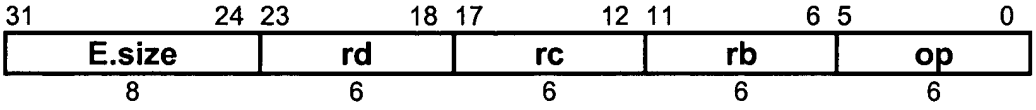

FIG. 54B

Definition

```
def mul(size,h,vs,v,i,ws,w,j) as
    mul ← ((vs&v_size-1+i)^h-size || v_size-1+i..i) * ((ws&w_size-1+j)^h-size || w_size-1+j..j)
enddef

def EnsembleInplace(op,size,rd,rc,rb) as
    if size=1 then
        raise ReservedInstruction
    endif
    d ← RegRead(rd, 128)
    c ← RegRead(rc, 128)
    b ← RegRead(rb, 128)
    case op of
        E.MUL.ADD, E.MULSUB, E.MUL.ADDC, E.MULSUBC:
            cs ← 1
            bs ← 1
        E.MUL.ADDM, E.MULSUBM:
            cs ← 0
            bs ← 1
        E.MUL.ADDU, E.MULSUBU:
            cs ← 0
            bs ← 0
    endcase
    h ← 2*size
    for i ← 0 to 64-size by size
        di ← d2*(i+size)-1..2*i
        case op of
            E.MUL.ADD, E.MUL.ADDU, E.MUL.ADDM:
                p ← mul(size,h,cs,c,i,bs,b,i) + di
            E.MUL.ADDC:
                if (i & size) = 0 then
                    p ← mul(size,h,cs,c,i,bs,b,i) - mul(size,h,cs,c,i+size,bs,b,i+size) + di
                else
                    p ← mul(size,h,cs,c,i-size,bs,b,i) + mul(size,h,cs,c,i,bs,b,i-size) + di
                endif
            E.MULSUB, E.MULSUB.U, E.MULSUB.M:
                p ← mul(size,h,cs,c,i,bs,b,i) - di
            E.MULSUBC:
                if i & size = 0 then
                    p ← mul(size,h,cs,c,i,bs,b,i) - mul(size,h,cs,c,i+size,bs,b,i+size) - di
                else
                    p ← mul(size,h,cs,c,i-size,bs,b,i) + mul(size,h,cs,c,i,bs,b,i-size) - di
                endif
        endcase
```

FIG. 54C

```
            z2*(i+size)-1..2*i ← p
        endfor
        RegWrite(rd, 128, z)
enddef
```

Exceptions

None

FIG. 54C (cont'd)